VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Mobilní aplikace pro podporu individuálních fitness programů

Fitness and Training Mobile Applications

2014 David Sklenář

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **David Sklenář**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Mobilní aplikace pro podporu individuálních fitness programů
Fitness and Training Mobile Applications

Zásady pro vypracování:

Cílem práce je vytvoření systému dle potřeb cílové skupiny, což jsou lidé zabývající se cvičením, fitness a zdravým způsobem života. Student získá a vyhodnotí požadavky na systém a na základě analýzy vytvoří návrh aplikace, která umožní komplexní práci s individuálními plány cvičení, kalkulátorem kalorií, ukládáním výsledků, milníků a statistikou zlepšující přehled o výsledcích uživatele. Výsledná aplikace bude podporovat víceuživatelský přístup (tvorbu účtů) a umožní přístup mobilním zařízením, které zjednoduší dostupnost aplikace při tréninku.

Specifické cíle práce jsou:
1. Vyhodnocení trhu a existujících aplikací na platformě iOS. Získání požadavků od cílové skupiny uživatelů.
2. Analýza a návrh aplikace společně s volbou vhodné kombinace technologií umožňující víceuživatelský přístup a podporu mobilních zařízení běžících na platformě Apple iOS.
3. Tvorba samotné aplikace na základě vytvořeného modelu.

Seznam doporučené odborné literatury:

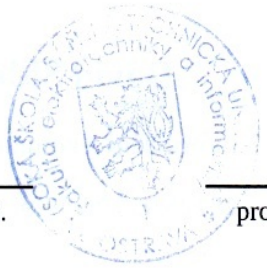
[1] KOCHAN, Stephen G. Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone. Vyd. 1. Brno: Computer Press, 2010, 550 s. ISBN 978-80-251-2654-7.
[2] Jeff LaMarche : iPhone SDK: Průvodce vývojem aplikací pro iPhone a iPod touch. Computer Press, 2010, ISBN 978-8-0251-2820-6.
[3] Kyle Richter: iOS Components and Frameworks: Understanding the Advanced Features of iOS 6 (Developer's Library), 2013, ISBN 978-0321856715 (zatím nevydáno)
[4] https://developer.apple.com/library/ios/navigation/

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michael Alexander Košinár**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014



doc. Ing. Miroslav Vozňák, Ph.D.	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7.5. 2014 .....

Rád bych na tomto místě poděkoval Ing. Michaelovi Alexandrovi Košinárovi za cennou zpětnou vazbu a odborné vedení během mé práce.

Abstrakt

Tato bakalářská práce se zabývá tvorbou mobilní aplikace pro podporu individuálních fitness programů. V úvodní části práce je vyhodnocen trh s aplikacemi a jsou zanalyzovány uživatelské požadavky, kde je brán ohled na potřeby cílové skupiny, kterou tvoří lidé zabývající se zdravým způsobem života a cvičením ve fitness centrech. Nastíněny jsou také požadavky nutné pro vývoj nad platformou iOS, na které je aplikace realizována. V další části práce je společně s vhodně zvolenou technologií navrženo řešení pro realizaci. V poslední části práce je podrobně rozebrána samotná implementace, kde jsou popsány frameworky a třídy s nimiž se v programu pracuje. Nedílnou součástí této práce je také výsledná spustitelná aplikace, která je obsažena v příloze na kompaktním disku.

Klíčová slova: iOS, mobilní aplikace, Objective-C, Core Data, iCloud, fitness, iPhone, bakalářská práce

Abstract

This bachelor thesis deals with the creation of a mobile application to support individual fitness programs. First part of thesis includes evaluation of market with existing applications and requirements analysis. Attention is focused to needs of the target group consisting of people interested in a healthy lifestyle and visitors fitness centers. Outlined are also requirements for the development in iOS platform on which the application is implemented. Second part contains software design of solution for implementation together with suitable technology. The author also describes most important frameworks and classes which are used in implementation and which the application operates with. An integral part of this work is the final executable application that is included in the appendix on compact disc.

Keywords: iOS, mobile application, Objective-C, Core Data, iCloud, fitness, iPhone, bachelor thesis

Seznam použitých zkratek a symbolů

SDK	–	Software Development Kit
IDE	–	Integrated Development Environment
HIG	–	Human Interface Guidelines
MVC	–	Model View Controller
REST	–	Representational State Transfer
API	–	Application Programming Interface
XML	–	Extensible Markup Language
HTTP	–	Hypertext Transfer Protocol
CRUD	–	Create, Read, Update, Delete
URL	–	Uniform Resource Locator
SQL	–	Structured Query Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Nacházíme se v době digitální. Málokdo si dokáže představit život bez moderních informačních technologií. Informační technologie zasahují nejen do dění ve všech odvětvích, ať už se jedná o lékařství, stavebnictví, ekonomii a mnoha dalších, ale především do běžného lidského života. Hlavní myšlenkou rozšíření tohoto oboru do firem a mezi širokou veřejnost bylo umožnit lidem pracovat pohodněji, rychleji a efektivněji. Výpočetní technika vytvořila cestu k novým způsobům komunikace mezi lidmi a práce s daty. Prakticky cokoli, co je dnes potřeba evidovat, se zaznamenává v převážné většině případů digitálně skrze softwarové programy a aplikace.

Nejnovějším trendem na trhu se softwarovými produkty je prezentovat uživateli data ještě pohodlnějším způsobem a to na mobilních zařízeních s dotykovou obrazovkou. Dotykové ovládání můžeme označit za velmi pohodlný způsob interakce mezi uživatelem a zařízením, kdy obrazovka už není chápána jen jako výstupní zobrazovač informací, jako tomu bylo v minulosti, nýbrž kombinuje vstupní funkci společně s výstupní. Díky mobilním zařízením jako jsou tablety nebo mobilní telefony, které tento způsob ovládání využívají, je umožněno mít svá data jako na dlani a pracovat s nimi je možné prakticky odkudkoliv a kdykoliv. Této skutečnosti se bude držet i tato práce.

Cílem práce je vytvořit mobilní aplikaci pro platformu iOS umožňující správu individuálních cvičebních programů, která umožní zapisování dosažených výsledků již při samotném tréninku. Budou objasněny základní požadavky pro vývoj aplikací nad platformou iOS, dojde k vyhodnocení trhu již existujících aplikací s podobnou tématikou a na základě analýzy požadavků cílové skupiny bude aplikace s ohledem na tyto požadavky navržena a následně implementována.

2 Vyhodnocení trhu

V současné době se na trhu s mobilními aplikacemi zabývající se fitness tématikou vyskytují řešení, která slouží uživatelům jako pomocník při dosahování jejich cílů. Hlavní funkcí, kterou disponují tyto aplikace, je možnost zaznamenávání dosažených výsledků u jednotlivých cviků. Dostupné jsou také aplikace specializující se na určitou svalovou partii jako např. břicho, ruce, hrudník, nohy apod. Množina se zúží, pokud se zaměříme na hotová řešení nabízející práci s jídelníčky s možností evidence a kalkulace přijatých jídel během dne. V následujících podkapitolách budou popsány aplikace, které se v současnosti nacházejí v App Storu. Hlavními parametry, podle kterých bude daná aplikace posuzována, jsou možnosti vytváření vlastních cviků i tréninků, zobrazení dodatečných informací o cviku, zpětné dohledání zaznamenaných dat při tréninku, práce s jídly při tvorbě jídelníčku, možnost synchronizace a možnost pracovat s aplikací bez nutnosti uživatelské registrace.

2.1 Fitness Pro

Aplikace umožňuje uživatelům vytvoření vlastního tréninku z databáze cviků, vložení vlastních cviků, které databáze neobsahuje. Použití aplikace není podmíněna registrací uživatele a stažení je možné zdarma. U každého cviku je k dispozici obrázek, popis provedení zvoleného cviku, obtížnost, primárně a sekundárně zapojené svaly. Aplikace s sebou přináší 3 předdefinované tréninky s možností vytvoření vlastních. Evidence zvednuté zátěže a počtu opakování pro každou sérii je zdlouhavé z důvodu použití tlačítek plus-minus, kdy každé zmáčknutí těchto tlačítek zvýší nebo sníží zátěž o 1 kg. Uživatel může zaznamenávat pro zvolené datum tělesnou váhu, tuk a míry jednotlivých partií jako jsou obvod pasu, hrudníku, stehen apod. Tohle je však možné až po přihlášení k Facebook účtu, kdy aplikace zároveň žádá o přístup k seznamu přátel, jinak tato funkce není povolena. Mezi placené funkce aplikace patří přístup k cvičebním videím správného provedení cviku a odstranění reklamního banneru. Co však aplikace neumožňuje je zobrazení odcvičených tréninků a zpětné dohledání dosažených výsledků v tréninkový den např. v kalendáři. Aplikace neobsahuje žádnou možnost tvorby jídelníčku, ani jednoduchý kalkulátor kalorií.

2.2 Fitlist

Registrace uživatele pro použití aplikace není potřeba a zdarma je k dispozici její omezená verze. Aplikace své uživatele zaujme především designem a celkovou přehledností. Stejně jako v předchozím případě obsahuje databázi cviků, avšak pouze jejich názvy. Obrázky cviků a jejich informace o tom, jakou svalovou skupinu cviky procvičují nebo jaké je správné provedení cviků se už uživatel nedozví. Čím však tato aplikace disponuje oproti předešlé je přehlednost a možnost zobrazení již odcvičených tréninků prostřednictvím kalendáře. Na druhou stranu neobsahuje fuknce pro výpočet kalorického příjmu ze zadaných jídel stejně tak, jako předchozí řešení. Fitlist nabízí nákup premiového vylepšení, obsahující zpřístupnění výchozích tréninků společně s vlastní tvorbou a zobrazením statistik. To vše je však možné až po zakoupení premiového vylepšení.

2.3 Fitness Buddy

Následující řešení poskytuje uživateli výchozí množinu cviků a 9 tréninků skládajících se z různých cviků podle zaměření tréninku na svalovou partii. Ve verzi aplikace, kterou lze z App Store stáhnout zdarma, je uživateli umožněno vytvořit si pouze 2 vlastní tréninky a to z výchozí databáze cviků, protože vkládat vlastní cviky je možné až v placené Pro verzi stejně tak jako zobrazení statistik. Zpětné dohledání zaznamenaných tréninků je možné skrze kalendář nebo časovou osu, která však zobrazí posledních 5 tréninků v případě aplikace zdarma. Mezi další placené funkce jež lze získat je měření hmotnosti člověka, mír jednotlivých tělesných partií či krevního tlaku. Možnost vložení potravin a následná tvorba jídelníčku možná není.

2.4 MyFitnessPal

Jedná se o aplikaci, která je dostupná volně ke stažení bez jakéhokoliv poplatku umožňující práci s jídelníčky. Uživatel se při prvotním spuštění aplikace musí zaregistrovat a následně zadat cíl, kterého chce dosáhnout. A to zda chce přibrat nebo zhubnout. Po zadání současné váhy uživatele je vypočítán doporučený kalorický příjem s ohledem na stanovený cíl v předchozím kroku. Do jídelníčku je umožněno vkládání potravin z databáze plus vytváření vlastních a k nim přiřazovat nutriční hodnoty. Celkově je aplikace zaměřená především pro práci s jídly a pro zaznamenávání denního kalorického příjmu a výdeje. Neobsahuje však tvorbu vlastních tréninků s možností zobrazení již odcvičených. Silové cviky lze zařadit do denních záznamů, neobsahují však popis správného provedení, obrázek ani informaci o procvičované svalové partii či vložení kalorického výdeje.

2.5 Shrnutí dostupných aplikací

Každá z aplikací má svá pro a proti. Všechny zmíněné aplikace a jejich výčet funkcí jsou zobrazeny v tabulce 1 pro lepší přehlednost. Cílem tak bude vytvořit aplikaci splňující všechny kritéria, jenž definuje levý sloupec tabulky. A ta jsou: zobrazení detailu cviku, možnost vytváření vlastních cviků i tréninků, zobrazení zaznamenaných dat v minulosti, synchronizace se serverem, práce s jídelníčkem a to vše bez přídavné registrace uživatele.

	Fitness Pro	**Fitlist**	**Fitness Buddy**	**MyFitnessPal**
Detail cviku	ANO	NE	ANO	NE
Vlastní cviky	ANO	ANO	poplatek	ANO
Vlastní tréninky	NE	poplatek	max. 2 (poplatek)	NE
Historie záznamů	NE	ANO	ANO	ANO
Synchronizace	NE	NE	ANO	ANO
Práce s jídly	NE	NE	NE	ANO
Nenutnost registace	ANO	ANO	ANO	NE

Tabulka 1: Funkce jednotlivých aplikací

3 Analýza a návrh aplikace

Před samotným návrhem aplikace bylo nutné získat od cílové skupiny uživatelské požadavky na aplikaci. Ty byly získány formou osobního rozhovoru od pravidelných návštěvníků fitness center a lidí zabývající se zdravým životním stylem a zanalyzovány byly z pohledu jejich vlastních potřeb. Tyto požadavky byly rozděleny na funkční a nefunkční.

3.1 Funkční požadavky

Mezi hlavní funkční požadavky, na které bude kladen důraz při realizaci aplikace patří:

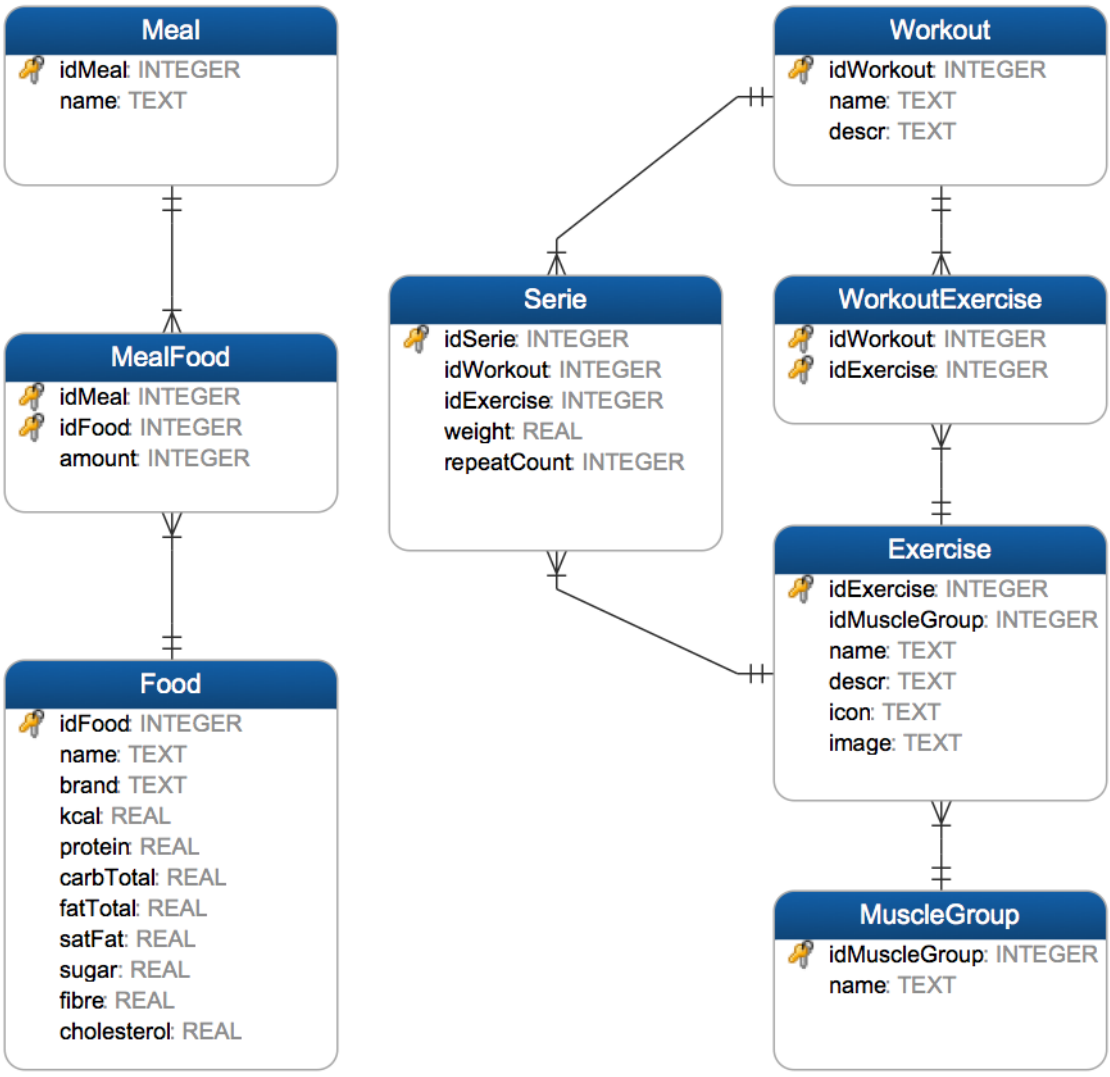
- uživatel bude moci vytvářet, editovat, mazat vlastní tréninky i cviky
- uživatel bude moci vytvářet, editovat, mazat dosažené výsledky při tréninku u jednotlivých cviků
- cviky budou seskupeny na základě svalových partií, které procvičují
- bude umožněno dohledání odcvičených tréninků
- uživatel bude moci zobrazit již odcvičené tréninky, jejich cviky i série
- bude umožněno zobrazit i dvoufázový trénink
- uživatel bude moci vytvářet, editovat, mazat jídla i suroviny
- uživatel bude moci přiřazovat množství přijatých potravin
- uživatel bude moci přiřazovat potraviny do jídel dle jejich doby příjmu
- každé jídlo bude poskládáno ze surovin
- u každé potraviny bude umožněno vložení nutričních hodnot
- uživateli aplikace nabídne spočtení celkového kalorického příjmu

3.2 Nefunkční požadavky

1. aplikace bude realizována nad iOS platformou
2. aplikace bude použitelná bez nutnosti registrace uživatele
3. práce s aplikací bude možná bez nutnosti přístupu k internetu
4. aplikace umožní synchronizovat data mezi více iOS zařízení
5. pro synchronizaci bude nutná jen minimální uživatelská režie

3.3 Datová analýza

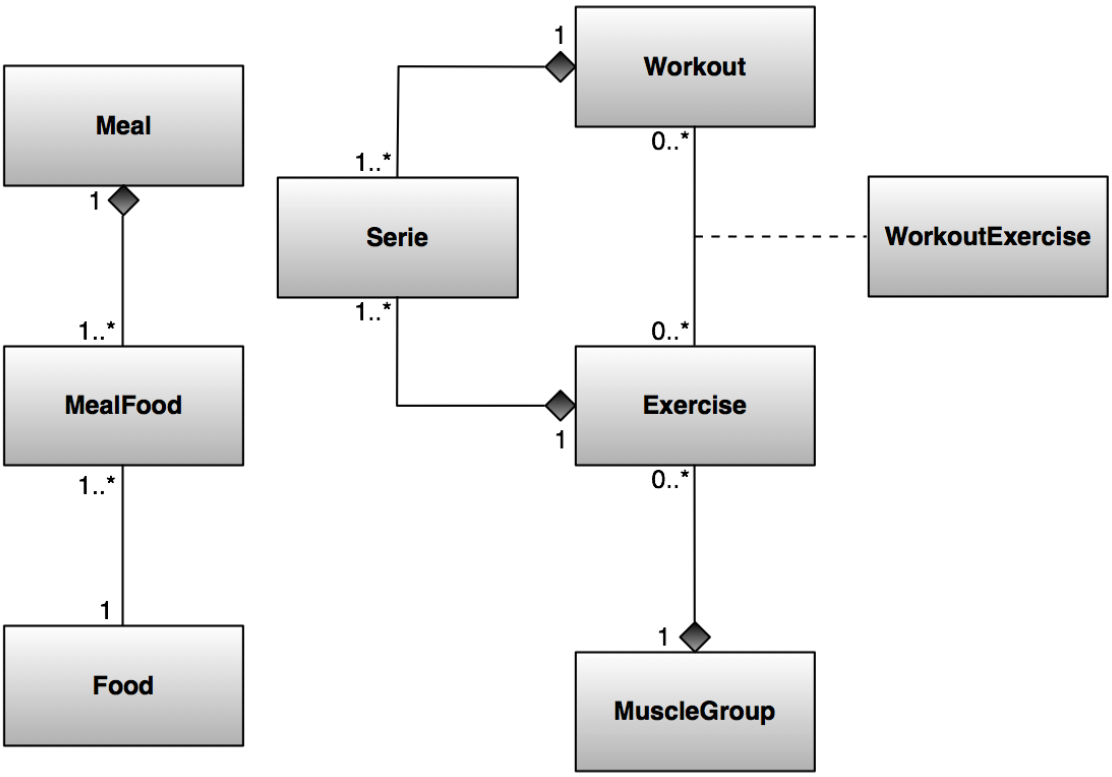
Po získání uživatelských požadavků byla provedena jejich analýza a následně navrhnut konceptuální datový model představující pohled na data z logické úrovně. Hlavními úkoly tohoto modelu je nalézt entity a vztahy mezi nimi společně s atributy entit, jež je potřeba evidovat. Konceptuální datový model je zobrazen na obrázku 1.



Obrázek 1: Výsledný diagram E-R modelu

3.4 Výstup analýzy

Výstupní analýzou je známo chování aplikace, kde lze tento výstup reprezentovat celkovým pohledem na systém, kde jsou známy třídy, jejich atributy a metody společně se vzájemnými vazbami mezi třídami. Tento výstup je reprezentován třídním diagramem znázorněným na obrázku 2.



Obrázek 2: UML třídní diagram aplikace

4 Implementační platforma

Protože se tato práce zabývá tvorbou aplikace pro platformu iOS, bude tato platforma na následujících řádcích popsána.

Platformou iOS (dříve iPhone OS) je nazýván operační systém používaný společností Apple v mobilních zařízeních iPhone, iPod Touch a iPad. Apple používá ve svých osobních počítačích operační systém OS X (dříve Mac OS X) a systém iOS je omezenou verzí tohoto systému. Tato odlehčená verze neobsahuje veškerou funkcionalitu jakou disponuje OS X, avšak přidává podporu dotykového ovládání, které na osobních počítačích nenajdeme. Oficiálně byla první verze systému iOS představena v roce 2007 společně s historicky první verzí iPhonu. Od té doby uběhlo několik let, kdy každý rok byla vydána nová verze, která s sebou přinesla různá vylepšení. Dnes je k dispozici verze s označením 7, u které proběhly poprvé po šesti letech změny týkající se vzhledu uživatelského rozhraní.

4.1 Vývoj pro iOS

K platformě iOS neodmyslitelně patří aplikace, které jsou dostupné ke stažení skze internetový obchod App Store, jež tvoří hodnotnou část zisků firmy Apple. Přistupovat k tomuto obchodu lze prostřednictvím programu iTunes nebo přímo z mobilního zařízení s iOS verze 2.0 nebo vyšší.

Vytvářet iOS aplikace znamená pro vývojáře jisté požadavky, ať už po stránce hardwarové, finanční či znalostní. Aplikace lze tvořit pouze na počítačích firmy Apple s operačním systémem OS X, píší se v programovacím jazyce Objective-C a doporučovaným vývojovým prostředím pro vývoj je Xcode. Hlavním frameworkem, který je pro tento účel použit je *Cocoa Touch*, jehož vybrané části jsou podrobněji popsány v kapitole Implementace. Dokumentace je dostupná z vývojového prostředí Xcode nebo pak na stránkách vývojářského portálu[1]. Znalost angličtiny je při vývoji takřka nutností vzhledem k tomu, že neexistuje aktuální česká literatura, protože nové verze iOS jsou vydávány v ročních cyklech, jak již bylo zmíněno v předchozí kapitole. Česká literatura je proto zastaralá a na trh se dostává se značným zpožděním. Vývojáři tak nezbývá nic jiného než sáhnout po literatuře v jiném jazyce.

4.2 Xcode

Jak už bylo zmíněno, Xcode je vývojové prostředí pro vytváření aplikací pro iOS a OS X. Tato aplikace je k dispozici ke stažení zdarma skrze App Store pro osobní počítače. Xcode balík obsahuje mnoho užitečných nástrojů a frameworků nutných při vývoji. Zde je jejich velmi stručný výčet:

- Xcode IDE
- OS X a iOS SDK

[1] `https://developer.apple.com/library/ios/navigation/`

- kompilátor
- debugger
- aplikaci Instruments
- iOS simulátor
- a mnoho dalších užitečných nástrojů, které usnadní práci při vývoji.

4.3 Apple developer program

Vývojář, jenž má v úmyslu testovat svou aplikaci na reálném zařízení nebo bude v budoucnu chtít distribuovat aplikaci na App Store, musí splnit určitá pravidla, která Apple nařizuje. Je nutné pořídit si vývojářský účet a zapojit se do některého z vývojářských programů. Vstoupení do vývojářského programu má však své klady i zápory:

- možnost testování aplikací na reálném zařízení (iPhone, iPod Touch, iPad)
- podpora testování aplikace pro iCloud v simulátoru (s příchodem iOS SDK 7)
- přístup k beta verzím iOS
- možnost umístit aplikace v App Store
- poplatek za členství 99 dolarů ročně
- 30% ze zisků aplikace náleží Applu v případě umístění placené aplikace.

4.4 Development Provisioning Profile

Testování aplikací na reálných zařízeních jako jsou iPhone, iPad či iPod Touch je možné až po vytvoření tzv. *vývojářského provizního profilu*. Tento profil dává vývojářům možnost svázat aplikaci se zařízením, a zároveň tím i umožnit spuštění aplikace na iOS zařízení. Zařízení, které je obsaženo v tomto profilu může být použito pro testování. Profil musí být nainstalován na každém zařízení, které se vývojář rozhodne pro testování používat. Registrovat je možné až 100 zařízení. Každý provizní profil obsahuje vývojářské certifikáty, unikátní identifikátor zařízení (UDID) a identifikátor aplikace (App ID). Vývojářský certikifát umožňuje podepsat jakoukoli aplikaci, pro kterou má vývojář k dispozici zdrojový kód. Aby bylo možné spustit aplikaci na reálném zařízení, musí tímto certifikátem disponovat jednak iOS zařízení, na kterém chceme aplikaci spustit, tak i počítač, ze kterého se aplikace instaluje.

Tento typ certifikátu však neumožnuje distribuci aplikace na App Storu. K umístění aplikace do toho internetového obchodu je nutné vytvořit *distribuční certifikát* a tzv. *distribuční provizní profil*. Vytváření profilů obou variant se provádí skrze webové rozhraní, které je dostupné registrovaným členům iOS vývojářského programu[2].

[2] `https://developer.apple.com/devcenter/ios`

4.5 Proces schválení

Protože je ze strany Applu brán ohled na to, aby aplikace umístěné v internetovém obchodě App Store dosahovaly jistých kvalit, doporučuje se řídit se dokumentem zvaným HIG[3]. Tento dokument obsahuje informace a doporučení o tom, jak by měly být v aplikaci správně rozvrženy prvky uživatelského rozhraní, aby nerušili celkový dojem, čistotu designu a s tím spojený prožitek uživatele při ovládání aplikace. Každá aplikace musí před vložením do obchodu projít schvalovacím procesem. Proces schválení aplikace ze strany Applu může nabývat od dnů až po týdny.

[3] `https://developer.apple.com/library/ios/documentation/userexperience/conceptual/MobileHIG/index.html`

5 Implementace aplikace

V následující kapitole bude popsána samotná tvorba aplikace, návrhové vzory využívané v aplikaci a pozornost bude také věnována frameworkům, knihovnám a prvkům uživatelského rozhraní použitých v aplikaci.

5.1 Návrhové vzory

Návrhové vzory se využívají při návrhu počítačových programů k popsání obecného řešení problémů, se kterými se může vývojář během psaní programu setkat. V této podkapitole je zmíněn výčet návrhových vzorů použitých v aplikaci.

5.1.1 MVC

Prvním návrhových vzorem nebo spíše architektonickým vzorem hojně využívaným při vývoji nejen iOS aplikací je Model-View-Controller. Základní myšlenkou architektury MVC je rozdělit zdrojový kód aplikace do tří logicky nezávislých sekcí a to na datový model, řídící logiku a uživatelské rozhraní. Komunikace mezi objekty Model a View by měla být vždy přes Controller nikoliv napřímo, aby případná změna některého z nich měla jen minimální dopad na ostatní sekce [1].

5.1.1.1 Model Sekce model reprezentuje datový model (někdy také označované jako jádro) aplikace. V případě této bakalářské práce budou model charakterizovat třídy vyplývající z výstupu analýzy z kapitoly 3.4.

5.1.1.2 Controller Řídící částí softwarové architektury MVC je Controller. Tato komponenta reaguje na události, které mohou v modelu nebo v uživatelském rozhraní nastat a na základě těchto údalostí provést určitou akci.

5.1.1.3 View Pohled představuje jednotlivé prvky uživatelského rozhraní, převádějící data reprezentovaná modelem do podoby, která je člověku přijatelnější pro konzumaci obsahu a více tak lahodí lidskému oku.

5.1.2 Delegát

Delegát je dalším častým návrhovým vzorem používaným při vývoji pro iOS platformu. Delegátem je chápán objekt zodpovědný za provádění určitých akcí pro jiné objekty. V aplikaci je tento návrhový vzor implementován hned v několika částech. Například mezi třídami `ExerciseMuscleGroupTVC` a třídou `AddExerciseTVC`, kdy je při zadání nového cviku potřeba vybrat procvičující svalovou skupinu, kterou je nutno vybrat z následující obrazovky a kliknutím na jeden z objektů ze seznamu je tato informace předána obrazovce předchozí [1].

5.1.3 Pozorovatel

Návrhový vzor Pozorovatel umožňuje komunikovat mezi objekty, které spolu nejsou těsně vázány. Objekt je schopen poskytovat informace jiným objektům aniž by o nich věděl jakékoli vlastnosti [2].

Tento návrhový vzor je v aplikaci použit právě tehdy, když nastane v databázi změna například v podobě přidání, editace nebo smazání tréninku na jednom ze zařízení. Při vložení nového tréninku bude mít první zařízení v databázi o jeden cvik navíc než databáze v zařízení druhém. V této situaci je vhodné druhé zařízení informovat o tom, že došlo k přidání tréninku v prvním zařízení a zajistit tak konzistenci dat všech databází na všech zařízeních. Poté, co se změnil obsah databáze je nutné informovat uživatelské rozhraní, aby došlo k aktualizaci uživatelského rozhraní vykreslující cviky do seznamu.

5.2 Synchronizace

Protože jedním z požadavků na aplikaci byla synchronizace uživatelských dat mezi více iOS zařízeními a s tím spjatá komunikace se serverem, bylo nutné před samotnou implementací aplikace na základě těchto požadavků vhodně zvolit (technologii) metodu pro přístup ke zdroji dat, se kterými bude aplikace pracovat.

Proč použít serverové úložiště? Serverové řešení pro uložení dat je vhodné použít pro možnosti synchonizace dat mezi více zařízeními. Další výhodu v použití tohoto typu úložiště lze shledat v zachování uživatelských dat i po smazání aplikace z iOS zařízení.

5.2.1 REST API

Jako první možností, která se nabízí pro daný problém je využití REST API. Tato architektura komunikace mezi klientem a serverem umožňuje přistupovat k datům dle zadané URL adresy. Komunikace probíhá skrze HTTP protokol, kdy klient vyšle HTTP požadavek směrem k serveru a na základě typu požadavku, zda chce klient získat, vložit, upravit nebo smazat určitá data se ve webové aplikaci provede nad databází mapování tohoto požadavku na akce známé pod zkratkou CRUD. Mapování dotazovacích metod protokolu HTTP je znázorněno v tabulce 2.

Akce, která se má provést určuje právě typ HTTP požadavku. Například pro získání kolekce dat nebo jen určitého objektu se vyšle požadavek metodou GET, který z CRUD operací odpovídá akci “Retrieve”, tedy získání dat z databáze jejím dotazováním výběrovým dotazem SELECT. Každá webová aplikace podporující REST komunikaci má jednoznačně definováno několik URL adres, nad kterými se tyto operace mohou provést a každá adresa znamená přístup k různým částem aplikace tudíž i datům.

5.2.2 iCloud

iCloud je pojmenování pro službu umožňující synchronizaci uživatelských dat napříč Apple zařízeními. Hlavní myšlenka použití iCloudu spočívá v umístění uživatelského obsahu

HTTP metoda	CRUD operace	SQL příkaz
PUT	Create	INSERT
GET	Retrieve	SELECT
POST	Update	UPDATE
DELETE	Delete	DELETE

Tabulka 2: mapování metod HTTP protokolu na CRUD operace a SQL příkazy

na serverech společnosti Apple, kdy uživatel může tato data měnit a synchronizovat. K dispozici má minimálně 5 GB prostoru a to pro různé druhy souborů od jednoduchých textových poznámek přes hudbu a obrázky po zálohy samotného operačního systému. iCloud je intergován v základních aplikacích jako je Kalendář, Kontakty, Poznámky, Připomínky, Mail, Safari, apod. K dispozici jsou i vývojářské nástroje, které mohou vývojáři do svých aplikací integrovat a využít tak potenciál této technologie.

Tím se vývojářům otevírá další možnost v komunikaci aplikace se serverem, kdy oproti jiným službám cloudového řešení spočívá výhoda iCloudu v tom, že je implementován na úrovní operačního systému, tudíž se uživatel pro použití aplikace využívající iCloud úložiště nemusí nikde registrovat. Vystačí si s Apple ID, které je nutné už jen pro samotné stažení aplikace z App Store. Implementací tohoto řešení se zabývá tato bakalářská práce.

5.2.2.1 Nasazení iCloudu

V minulosti bylo vývojářům umožněno testovat aplikace využívající iCloud technologii jen mezi reálnými zařízeními, což kladlo na vývojáře zvýšené nároky na vývoj z hlediska finanční stránky. Až s příchodem nové verze iOS 7 a nové vývojářské sady, která byla oficiálně vypuštěna v září roku 2013, se situace zlepšila. Od této verze je možné testovat iCloud technologii v simulátoru, avšak pro názorné otestování synchronizace dat v reálném čase se doporučuje použití alespoň jednoho reálného zařízení v kombinaci se simulátorem. S novou verzí SDK bylo vydáno také nové API pro integrování iCloud technologie do aplikací. Pro nasazení samotné technologie a její testování v aplikaci je nutné být registrovaným členem některého z vývojářských programů. Samotné nastavení iCloudu pro vyvíjenou aplikaci je nutno provést v *provizním profilu* a ve vývojovém prostředí Xcode.

Protože se jedná o relativně neprobádanou oblast v rámci akademických prací, budu se implementací tohoto řešení zabývat v rámci praktické části této bakalářská práce.

5.3 Uchovávání dat

Při práci s aplikací se mění stavy objektů podle toho, jak uživatel s aplikací pracuje. Tyto objekty je vhodné nějakým způsobem uložit, aby uživatel například po restartu zařízení nacházel aplikaci v takovém stavu, v jakém ji zanechal. Uživatele by nejspíš nepotěšilo, kdyby po pracném zaznamenávání výkonů, kterých během několika tréninků dosáhl, zjistil, že veškerá jeho práce s evidencí přišla nazmar. Z tohoto důvodu je vhodné ukládat tato

data, aby se předcházelo těmto nepříjemným situacím. V Cocoa Touch existuje několik možností, jak čelit této problematice a v další části práce budou uvedeny některé z nich.

5.3.1 Plist

Jednou z možností je ukládat objekty do textových souborů jako jsou *seznamy vlastností*. Tyto soubory jsou vývojářům známé jako tzv. plisty (z angličtiny *property list*), které jsou svou strukturou velmi podobné XML souborům. Uchovávání dat v těchto souborech se hodí kupříkladu pro nastavení aplikace, nicméně pro uložení rozsáhlých datových struktur není toto řešení příliš vhodné.

5.3.2 Relační databáze SQLite

Ideálním způsobem by bylo uchovávat veškerou datovou část aplikace uvnitř jednoho souboru. A to v databázi, kdy tato možnost zajistí patřičné vztahy mezi objekty a práce s ní bude pro programátora mnohem přijatelnější než neefektivní práce s plisty. Součástí vývojářské sady je i knihovna obsahující relační databázový systém SQLite, který lze využít pro problematiku uchování dat. Nicméně s rozsáhlým datovým modelem aplikace vzroste počet i rozsah SQL dotazů potřebných pro jeho pokrytí. Z tohoto důvodu by programátor ocenil možnosti využití nějakého druhu frameworku, který by mu práci s databází usnadnil. Řešením je framework známý jako Core Data.

5.4 Core Data

Core Data je framework umožňující vývojáři zefektivnit proces při interakci s databází. Jedná se o hojně využívaný framework mezi vývojáři, který je dostupný jak pro platfomu iOS, tak pro OS X. Framework však sám o sobě není databáze. Na místo psaní SQL kódu dovoluje programátorovi pracovat s objekty databázového modelu napřímo. Veškerý SQL kód spojený s objekty, který by musel být dodatečně napsán má v režii tento framework a vývojář tak nemusí investovat čas potřebný pro psaní a testování SQL dotazů a jejich optimalizaci. Použitím frameworku se tak sníží čas potřebný pro vývoj aplikace.

5.4.1 Perzistence dat

Princip uchování dat s využitím Core Data frameworku spočívá ve vytvoření objektového grafu, který z velké části odpovídá konceptuálnímu datovému modelu získaného po analýze uživatelských požadavků a následného návrhu implementace. Vývojář specifikuje entity, atributy jejich datové typy, vztahy mezi entitami, kardinalitu vztahů a další vlastnosti jakých by měla databáze nabývat. Datový model je možno definovat v grafickém editoru, jenž je součástí vývojového prostředí Xcode. Pokud se vývojář rozhodne ve svém projektu Core Data využít, znamená to pro něj implementaci tříd umožňující práci s tímto frameworkem.

5.4.1.1 NSManagedObjectModel Třída reprezentující objektový graf, kterému odpovídá databázový model. Vytvoření tohoto modelu probíhá v samotném vývojovém prostředí Xcode, kde se současně definují entity, vztahy mezi nimi apod. Nastavení datového modelu aplikace se provádí v souboru s koncovkou *.xcdatamodel*.

5.4.1.2 NSManagedObject Každá entita definovaná v datovém modelu aplikace odpovídá třídě, jejíž rodičem je právě třída `NSManagedObject`. Instance této třídy pak odpovídají objektům jedné konkrétní entity. Pro zjednodušení si lze objekt této třídy představit jako jeden konkrétní řádek tabulky. Atributy třídy (instanční proměnné) odpovídají atributům definovaných v objektovém grafu a přístup k nim je možný jako u každé jiné třídy.

5.4.1.3 NSManagedObjectContext Třída zaobalující objekty určené k uchování do oblasti, kterou si lze představit jako jakýsi kontejner nebo také kontext. Uvnitř tohoto kontextu se nacházejí objekty třídy `NSManagedObject` odpovídající datům uložených v databázi. K uchování objektu je nejprve nutné jej vložit do kontextu. To se provede zavoláním metody `insertNewObjectForEntityForName:inManagedObjectContext:`, která vrací referenci na vložený objekt s nímž lze v programu dále pracovat a nastavit tak případné hodnoty atributů. K samotnému uložení dat do databáze dojde až po zavolání metody `save:`, kterou je vhodné použít v kombinaci s metodou `hasChanges` detekující změny v kontextu. Vložení objektu do databáze je znázorněno ve výpisu 1.

5.4.1.4 NSPersistentStoreCoordinator Třída zodpovědná za vytvoření a přístup k úložišti dat. Určuje umístění, kde budou data uložena, nastavení, konfiguraci modelu a spojení kontextu se způsobem, jakým budou data uložena.

5.4.1.5 NSPersistentStore Třída zodpovědná za to, v jakém formátu budou data uložena. Možnými formáty pro perzistentní uložení dat je formát XML, binární soubor nebo SQLite databáze. Pro systém iOS je výchozím a také nejvhodnějším řešením právě databáze. XML úložiště pro iOS není dostupné.

```
Serie *newSerie = [NSEntityDescription
    insertNewObjectForEntityForName:@"Serie"inManagedObjectContext:[
    self managedObjectContext]];
// for conversion NSString to NSNumber
NSNumberFormatter *f = [[NSNumberFormatter alloc]init];
[f setNumberStyle:NSNumberFormatterDecimalStyle];

[newSerie setWeight:[f numberFromString:serieWeightTextField text]];
[newSerie setRepeatCount:[f numberFromString:[
    serieRepeatCountTextField text]];
[newSerie setDate:selectedDate];
```

```
[newSerie setExercise:selectedExercise];
[newSerie setWorkout:selectedWorkout];

NSError *error;
if([[self managedObjectContext] hasChanges]) {
    if([[self managedObjectContext] save:&error]) { // if save failed
        NSLog(@"Save failed: %@", [error localizedDescription]);
    } else {
        NSLog(@"Save succeeded");
    }
} else {
    NSLog(@"No changes in MOC - Nothing to save");
}
```

Výpis 1: Uložení série cviku po stisku tlačítka

5.4.2 Získávání dat

V následující části práce budou popsány hlavní třídy potřebné pro získání objektů z databáze. Obecně se získání objektů praktikuje pomocí požadavku, který je zavolán nad kontextem datového modelu s předem určenými parametry určující dodatečná omezení a typ výsledných objektů.

5.4.2.1 NSFetchRequest Tuto třídu je potřeba použít pro vytvoření požadavku, který chceme vykonat nad samotnou databází. Používá se při každé potřebě získat data z databáze.

5.4.2.2 NSEntityDescription Tato třída reprezentuje entitu databáze a tím i zároveň typ navrácených objektů. Proto je objekt této třídy nutno uvést při každém vykonání požadavku nad kontextem.

5.4.2.3 NSSortDescriptor Objekt této třídy určuje pořadí, v jakém budou objekty vráceny.

5.4.2.4 NSPredicate Objekt, jenž je potřebný pro nastavení omezujících pravidel, na základě kterých budou výsledná data filtrována.

Před vykonáním požadavku je nutno požadavek inicializovat a to nastavením výše uvedených vlastností, z nichž povinným parametrem je určení entity výsledných objektů při použití metody `executeFetchRequest:error`: pro vykonání požadavku nad kontextem. Návratovou hodnotou je pole objektů specifikovaných při inicializaci požadavku.

Často je v aplikaci nutno zobrazit seznam dat seřazených podle konkrétního klíče. Což by nebyl problém. Vždy, když by bylo potřeba tento seznam zobrazit, by se data uložila

do pole a následně použila jako zdroj dat pro seznam. Avšak při změně dat v seznamu by bylo potřeba učinit opatření pro správné zobrazení řádků seznamu a zdrojem dat, aby nedošlo k nekonzistenci mezi těmito objekty. `NSFetchedResultsController` je třída, jež splňuje tento účel a proto byla použita.

5.4.2.5 NSFetchedResultsController Jak už název napovídá se tato třída používá k efektivní práci s objekty po provedení požadavku pro získání dat z databáze. Třída poskytuje atribut pole s názvem *fetchedObjects*, kterému se po zavolání metody `performFetch:` přiřadí data korespondující s hodnotami zobrazenými v seznamu. Před vykonáním požadavku pro získání dat z databáze je nutné nastavit inicializační hodnoty instančních proměnných objektu `NSFetchedResultsController` z nichž je povinné zvolit entitu a alespoň jeden atribut entity, podle kterého budou data setřízena. Ve výpisu 2 je tato inicializace uvedena.

K řízení změn, ke kterým může během prací se seznamem dojít, je výhodné určit delegáta, který implementuje protokol `NSFetchedResultsControllerDelegate` pro notifikaci případných změn. Protokol definuje několik metod určující, co se má se seznamem stát například při přidání nového řádku nebo smazání řádku již existujícího.

```
// ...
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
NSManagedObjectContext *context = [self managedObjectContext];
NSEntityDescription *entity = [NSEntityDescription entityForName:@"
    Exercise" inManagedObjectContext:context];
[fetchRequest setEntity:entity];
NSSortDescriptor *mgSD = [[NSSortDescriptor alloc] initWithKey:@"
    muscleGroup.name" ascending:YES];
NSSortDescriptor *exerciseNameSD = [[NSSortDescriptor alloc]
    initWithKey:@"name" ascending:YES];
NSArray *sortDescriptors = [[NSArray alloc] initWithObjects: mgSD,
    exerciseNameSD, nil];
[fetchRequest setSortDescriptors:sortDescriptors];
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"ANY
    workouts = %@", selectedWorkout];
[fetchRequest setPredicate:predicate];
fetchedResultsController = [[NSFetchedResultsController alloc]
    initWithFetchRequest:fetchRequest managedObjectContext:context
    sectionNameKeyPath:nil cacheName: nil];
// ...
```

Výpis 2: Inicializace NSFetchedResultsControlleru pro získání pole cviků zvoleného tréninku

5.5 Manipulace s obrazovkami

Aplikace pro iOS jsou v převážné většině tvořeny více obrazovkami zobrazující různý obsah. Mezi těmito obrazovkami jsou definovány přechody, které jsou provedeny na základě akce provedené uživatelem. Akcí se rozumí provedení bloku kódu po předchozí interakci uživatele s prvkem uživatelského rozhraní. Jako akci lze považovat například stisk tlačítka nebo dotyk v předem definové oblasti obrazovky obecně. Tyto obrazovky je možné tvořit buďto přímo jen z kódu nebo použitím kódu společně s tzv. *Storyboardem*.

5.5.1 Storyboard

Storyboard slouží pro vizuální reprezentaci sítě obrazovek a přechodů mezi nimi, jakých může aplikace nabývat. Použitím storyboardu získá vývojář větší přehled nad jednotlivými obrazovkami aplikace a zároveň redukuje množství kódu potřebného pro jejich řízení. Ve storyboardu je definována také počáteční obrazovka, která se zobrazí po spuštění aplikace. Storyboard může být součástí projektu, kde celý strom obrazovek je definován v souboru s příponou *.storyboard*, kde jednotlivé obrazovky si lze představit jako vrcholy grafu a přechody mezi obrazovkami jako hrany grafu.

5.5.2 UIStoryboardSegue

Tato třída je zodpovědná za vykonání vizuálního přechodu mezi dvěma obrazovkami. Každý přechod obsahuje zdrojovou a cílovou obrazovku. Když je přechod spuštěn, ale animace přechodu ještě nenastala, zavolá se metoda `prepareForSegue:sender:`, která se používá pro předání potřebných dat cílovému controlleru. Toto chování je v aplikaci implementováno například v situaci, kdy je po stisknutí řádku v seznamu zobrazující tréninky, předána reference zvoleného tréninku další obrazovce zobrazující všechny cviky, jenž do vybraného tréninku spadají. Každý přechod je definován jednoznačným identifikátorem, aby bylo možné v kódu rozlišit více přechodů pro jednu obrazovku.

5.5.3 UIViewController

Každá obrazovka je z kódu řízena třídou `UIViewController`, která se stará o to, co bude na obrazovce zobrazeno. Do tohoto controlleru je možné umístit prvky uživatelského rozhraní a vytvořit tak vlastní vzhled dané obrazovky. To je možné s pomocí storyboardu nebo rovnou z kódu. Protože mnoho aplikací obsahuje více obrazovek, bylo by vhodné poskytnout uživateli mechanismus pro přehledné procházení aplikace s více obrazovkami. V Cocoa Touch je k dispozici více typů ViewControllerů, z nichž je k tomuto účelu vyhrazena třída `UINavigationController`. V následující části budou popsány ty, které jsou ve výsledné aplikaci implementovány.

5.5.3.1 UINavigationController Tento typ ViewControlleru používá pro zobrazení jednotlivých obrazovek tzv. *navigační zásobník*, který je implementován jako pole obrazo-

vek. První ViewController je v zásobníku umístěn na nultém indexu a odpovídá tedy kořenovému ViewControlleru. Poslední ViewController v zásobníku reprezentuje aktuálně zobrazenou obrazovku na displayi. Pro potřebu zobrazení nové obrazovky lze využít metodu `pushViewController:animated:`, která přidá na vrchol zásobníku ViewController předaný jako parametr této metody nebo lze zobrazení nové obrazovky definovat přechodem ve storyboardu. Naopak k zobrazení předchozí obrazovky lze z kódu řídit metodou `popViewControllerAnimated:`. Někdy je v aplikaci nutné zobrazit ViewController, který se nachází na samotném začátku zásobníku nebo v jeho určité části. Pro zobrazení počáteční obrazovky je na místě použít metodu `popRootViewControllerAnimated:`, která je v ukázkové aplikaci použita po vložení jídla do jídelníčku.

5.5.3.2 UITabBarController Implementací tohoto controlleru je v aplikaci zobrazen pruh s ikonami určující zobrazení jedné konkrétní obrazovky. Každý tab reprezentuje ViewController, kdy po výběru daného tabu je zobrazena počáteční obrazovka z hierarchie controllerů. V ukázkové aplikaci je tento typ controlleru implementován v kombinaci s navigačním controllerem, kdy každý tab ve spodní liště má svůj *navigační zásobník* ViewController, jenž je předem specifikován. Oba typy controllerů jsou specifikací třídy UIViewController, tudíž každý typ má svá specifika popsaná v předchozích dvou částech práce.

5.6 Prvky uživatelského rozhraní

Následující podkapitola bude věnována práci s vybranými prvky uživatelského rozhraní. Tedy s prvky, které je možné na obrazovce vidět a interagovat s nimi dotykem prstu. Komponenty jsou součástí frameworku známého jako UIKit, kde třída `UIView` je nadřazenou třídou všech viditelných prvků, které může uživatel ovládat.

5.6.1 UITableView

Nejčastějším prvkem, který bývá obsažen téměř v každé aplikaci je seznam (v některých literaturách také označovaný jako tabulka). Tato komponenta umožňuje uživateli zobrazit nejen textová data ve formě přehledného seznamu. Jednou z výhod použití tohoto prvku spočívá v možnosti vlastní konfigurace vzhledu. Díky tomu je možné definovat, jak budou jednotlivé řádky seznamu vypadat a přizpůsobit si tak vzhled komponenty samotné. Seznamy mohou být statické nebo dynamické. Seznamy mají vždy jeden sloupec, kdy počet řádků je omezen jen objemem dostupné paměti [3] Mohou obsahovat i sekce, seskupující objekty na základě specifikované společné vlastnosti. V aplikaci jsou použity seznamy jak se sekcemi, tak seznamy bez sekcí, seznamy se statickým i dynamickým obsahem.

Pro použití seznamu nebo-li tabulky je v aplikaci nutno použít objekt třídy `UITableView`. Tento objekt je zodpovědný za reprezentaci dat v jejich vizuální podobě. Každý seznam je tvořen řádky nebo-li buňkami, které definuje třída `UITableViewCell`.

5.6.1.1 UITableViewController

Pro zobrazení seznamu zabírající celou plochu obrazovky je vhodné použít třídu `UITableViewController`, která je k tomuto účelu přizpůsobena. Třída obsahuje proměnnou *tableView*, což je právě seznam, do kterého se budou data zobrazovat. Velikost zobrazených dat je omezená pouze dostupnou operační pamětí. Objekt třídy `UITableViewController` odpovídá protokolům `UITableViewDataSource` a `UITableViewDelegate`.

UITableViewDataSource Každý seznam potřebuje pro zobrazení dat vědět, jaká data se mají v jednotlivých řádcích zobrazit. Seznamy získávají data popisující jejich konfiguraci z objektu, který odpovídá tomuto protokolu [3]. Tedy z ViewControlleru, který řídí zobrazení obsahu na obrazovce displaye. Tento protokol definuje metody, které jsou implementovány při každém použití dynamického seznamu v aplikaci. Těm nejčastěji používaným bude věnováno pár řádků:

- `numberOfSectionsInTableView:` metoda se dotáže zdroje dat pro počet sekcí, které se mají v seznamu zobrazit a tuto hodnotu vrací. Metodu není nutno uvést, pokud seznam disponuje jedinou sekcí.
- `tableView:numberOfRowsInSection:` metoda se volá za účelem zjištění počtu řádků v dané sekci. Návratovou hodnotou je tedy počet řádků dané sekce.
- `tableView:tableView titleForHeaderInSection:` při použití seznamu používající sekce k oddělení obsahu je v aplikaci použita tato metoda, kdy pro zobrazení názvů jednotlivých sekcí je z parametru metody *section* zjištěno číslo sekce a k ní je přiřazen řetězec, který sekci odpovídá. Prosté seznamy (bezsekční seznamy) tuto metodu neimplementují z důvodu výchozího nastavení proměnné *section*.
- `tableView:cellForRowAtIndexPath:` zajímavější je až tato metoda, která se volá vždy, když je potřeba vykreslit jeden konkrétní řádek v seznamu. To, v jaké sekci a na jakém místě sekce se má řádek vykreslit určuje parametr *indexPath*, jenž zabaluje informaci o sekci i řádku do jednoho objektu. Metoda se volá nejen při počátečním zobrazení seznamu, ale také při každém posunu řádků jakýmkoli směrem, kdy je potřeba zobrazit řádky, které se na obrazovku nevešly.
- `tableView:commitEditingStyle:forRowAtIndexPath:` metoda je v aplikaci použita na několika místech za účelem smazání tréninku, cviku, jídla, apod. Smazání objektu v seznamu je uživateli umožněno gestem *swipe-to-delete*. Toto gesto je na na displayi rozpoznáno tahem prstu zprava doleva, přičemž se po uvolnění prstu uživateli zobrazí tlačítko pro smazání řádku, na kterém bylo gesto provedeno.

UITableViewDelegate Protokol definuje volitelné metody umožňující delegátovi řídit výběry, nastavení záhlaví a zápatí sekcí, případné smazání řádků, jejich změnu pořadí a další akce. U tohoto protokolu je nutné se zmínit o nejčastěji používané metodě, které tento protokol definuje.

- `tableView:didSelectRowAtIndexPath:` tato metoda detekuje stisknutí řádku a volána tedy je po jeho každém stisknutí. V aplikaci je implementována hned v několika třídách. Metoda je v aplikaci použita v místě, kdy místo toho, aby uživatel nemusel přidávat cviky do tréninku po jednotkách, je uživateli umožněno hromadné přidání cviků do tréninku označením více řádků. Výpis kódu 3 popisuje označení několika řádků seznamu, kde každý řádek reprezentuje konkrétní cvik. Každému označenému cviku je v seznamu zobrazen znak zaškrtnutí pro informaci uživatele. Po stisknutí tlačítka *Save* v navigační liště obrazovky se vybrané cviky do tréninku uloží.

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(
    NSIndexPath *)indexPath {
     Exercise *exercise = [self.fetchedResultsController
    objectAtIndexPath:indexPath];
     [self.view setTintColor:UIColorFromRGB(0x0066CC)];
     UITableViewCell *cell = [tableView cellForRowAtIndexPath:
    indexPath];
     if (cell.accessoryType == UITableViewCellAccessoryCheckmark) {
         cell.accessoryType = UITableViewCellAccessoryNone;
         [selectedObjects removeObject:exercise];
     } else {
         cell.accessoryType = UITableViewCellAccessoryCheckmark;
         [selectedObjects addObject:exercise];
     }
     [tableView deselectRowAtIndexPath:indexPath animated:YES];
}
```

Výpis 3: Řešení pro označení více cviků dotykem prstu

5.7 Knihovny třetích stran

Jedním z požadavků na aplikaci bylo umožnit uživateli dohledání dosažených výsledků, které byly do aplikace zařazeny. Jedním z řešení, jak tomuto problému čelit, bylo zobrazit uživateli chronologické zobrazení sérií v podobě seznamu. Tento přístup by pro uživatele nebyl zrovna přehledný a práce se seznamem by tak byla zdlouhavá a neefektivní. Lepší řešení by bylo zobrazit uživateli odcvičený trénink na základě dne, který by si uživatel sám zvolil. K tomuto řešení je vhodné použít kalendář.

Apple však ve své vývojářské sadě žádný kalendář nenabízí, a tak vývojáři nezbývá nic jiného než sáhnout po náhradní alternativě řešení a tou je použití knihovny třetích stran.

5.7.1 Tapku

V aplikaci je použita knihovna s názvem *Tapku*, což je knihovna napsaná pod frameworky Cocoa a UIKit jejíž autorem je Devin Ross. Záměr vzniku knihovny je poskytnout vývojáři přídavné ovládací prvky uživatelského rozhraní, které standardní vývojářská sada dodávaná Applem neobsahuje. Knihovna disponuje mnoha komponentami, které může vývojář ve svých aplikací implementovat. Prvky, které knihovna obsahuje shrnuje následující stručný výčet:

- *Coverflow* - v dřívějších verzích iOS byl tento prvek používán pro listování hudebních alb v nativním přehrávači hudby
- *Loading HUDs* - prvky určené např. pro informování uživatele o postupu načítání souborů
- *Slide to unlock* - tento prvek je všem majitelům iOS zařízení známý, protože se používá pro odemknutí telefonu gestem
- *Empty Sign* - prvek používaný pro reprezentaci prázného obsahu
- měsíční a týdenní kalendář
- upozorňující prvky
- klávesnice s vlastním uspořádání kláves
- přizpůsobitelné buňky seznamů, tlačítek a mnoha dalších komponent

5.7.1.1 Kalendář V aplikaci je pro zobrazení zaznamenaných dat použit kalendář a to konkrétně pro zobrazení odcvičených tréninků, kdy po zvolení data se uživateli zobrazí seznam tréninků odcvičených daný den. Po rozklepnutí tréninku je uživateli zobrazen seznam odcvičených cviků a zvolením cviku se uživatel zobrazí seznam sérií, kde u každé série je zaznamenána váha společně s počtem opakování.

Třída reprezentující samotný kalendář je `TKCalendarMonthView`, ale protože chceme zobrazit kalendář a pod ním seznam, do kterého se budou zobrazovat odcvičené tréninky, je k tomuto účelu dobré sáhnout po třídě `TKCalendarMonthTableViewController`.

TKCalendarMonthTableViewController Třída vytváří objekt, který řídí zobrazení kalendáře a seznamu nacházející se pod ním. `TKCalendarMonthViewController` je rodičovskou třídou obsahující referenci na objekt `TKCalendarMonthView`, což je právě zmiňovaný kalendář [6]. Třída `TKCalendarMonthTableViewController` obsahuje atribut *tableView* typu `UITableView`, takže reference na seznam je zajištěna také. Součástí inicializace kalendáře je vhodné nastavit časové pásmo a první den týdne metodou `initWithSundayAsFirst: timeZone:`, jinak se může stát, že bude kalendář po označení dne vracet špatný den.

5.7.1.2 Zobrazení tréninkových dnů Aby uživatel nemusel provádět výběr data z kalendáře naslepo a hledat tak, zda se náhodou v určitý den nenachází odcvičený trénink, je každý tréninkový den označen v kalendáři kolečkem. Toto označení bylo potřeba implementovat v metodě `calendarMonthView: marksFromDate:toDate:`, jenž je součástí protokolu `TKCalendarMonthViewDataSource`.

- `calendarMonthView:marksFromDate:toDate:` Metoda slouží k označení dnů v kalendáři, u kterých se má zobrazit značka, jenž dává uživateli najevo, že se pod daným dnem skrývá v minulosti odcvičený trénink. Tato metoda vrací pole objektů, kde dny se značkou odpovídají v poli prvku s logickou jedničkou a prvky bez značky v kalendáři, odpovídají prvkům s logickou nulou. Velikost pole záleží na měsíci, který uživatel v kalendáři zvolil. Protože může v kalendáři nastat situace, kdy jsou v kalendáři zobrazeny nejen dny zvoleného měsíce, ale i dny měsíce přechozího nebo následujícího, jsou do tohoto pole zahrnuty i dny z předchozího nebo následujícího měsíce.

  Dny s tréninkem jsou tedy v poli reprezentovány logickou jedničkou a dny bez tréninku logickou nulou. Uživatel tak může vidět tréninkové dny posledních dnů přechozího měsíce a zároveň část tréninkových dnů následujícího měsíce. Počet dnů zobrazeného měsíce a tedy i velikost pole je dán rozdílem dat předaných v parametrech metody.

  Protože série obsahuje informaci o datu, kdy byla provedena a zároveň i informaci, do jakého spadá tréninku je nutné získat série, které vyhovují rozsahu dat pro zvolený měsíc z parametrů *startDate* a *lastDate*. To má na starosti metoda `getSeriesFromDate:toDate`, která vrací pole sérií zadaných dnů, které používá metoda `generateDotsToCalendarWithStartDate:endDate:` vytvářející pole logických jedniček a nul, kde toto pole slouží jako zdroj dat pro označení dnů v kalendáři. Metoda `generateDotsToCalendarWithStartDate:endDate:` je popsána výpisem 4.

5.7.1.3 Zobrazení tréninků Když už uživatel vidí tréninkové dny, zbývá uživateli zobrazit tréninky v seznamu na základě zvoleného dne v kalendáři. K tomuto účelu byla použita metoda `calendarMonthView:didSelectDate:`, jenž je deklarována v protokolu `TKCalendarMonthViewDelegate` a volána je vždy při označení libovolného data v kalendáři. Získání pole tréninků je provedeno tak, že se nejdříve získá pole sérií daného dne a z těchto sérií se získají všechny tréninky, které se tento den cvičily. Tím je zajištěno zobrazení i dvoufázového tréninku.

Pole s tréninky je následně předáno seznamu jako zdroj dat pro jejich zobrazení. Reference na seznam je součástí třídy `TKCalendarMonthTableViewController`, která implementuje protokoly `UITableViewDelegate` a `UITableViewDataSource`, takže lze se seznamem pracovat jako to bylo popsáno v kapitole 5.6.1.

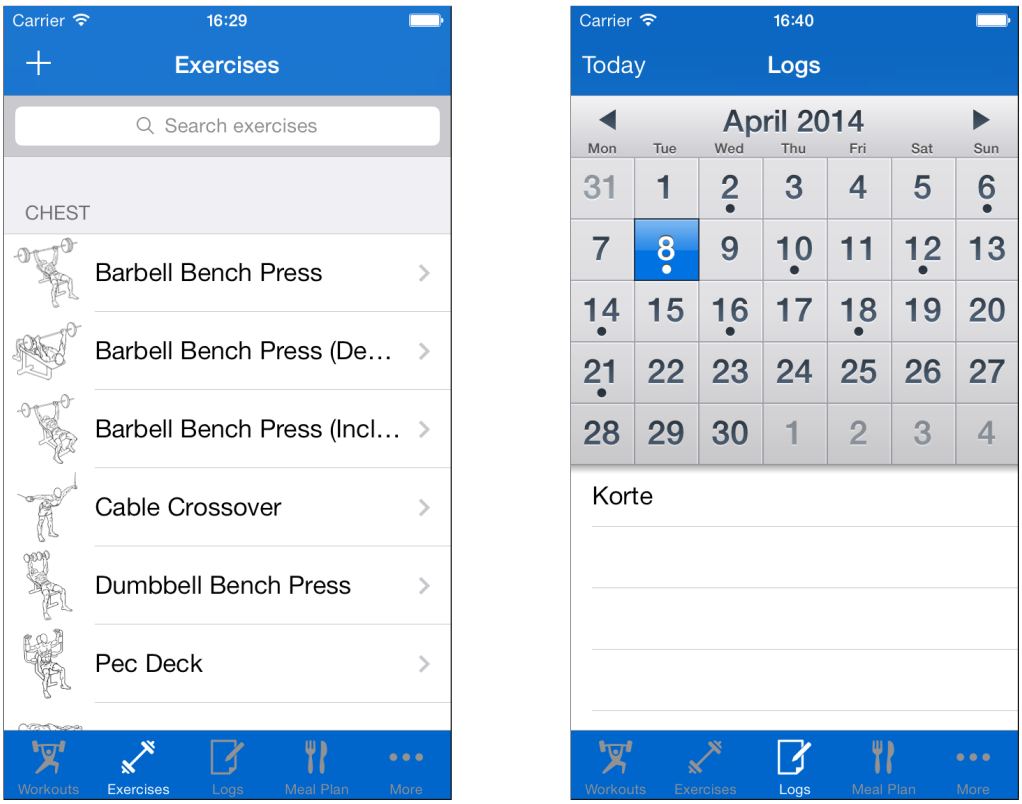
```
- (NSArray *)generateDotsToCalendarWithStartDate:(NSDate*)start
    endDate:(NSDate*)end{

  NSArray *series = [self getSeriesFromDate:start toDate:end];
  // getting days where exists some log
  NSMutableSet *dates = [[NSMutableSet alloc] init];
  for (Serie *s in series) {
    NSCalendar *calendar = [[NSCalendar alloc]
   initWithCalendarIdentifier:NSGregorianCalendar];
    calendar.timeZone = [NSTimeZone timeZoneWithAbbreviation:@"GMT"
   ];
    NSDateComponents *components = [calendar components:
   NSYearCalendarUnit|NSMonthCalendarUnit|NSDayCalendarUnit
   fromDate:s.date];
    [components setHour:00];
    NSDate *modifiedDate = [calendar dateFromComponents:components];
    [dates addObject:modifiedDate];
  }

  NSDate *dateIterator = start;
  NSMutableArray *daysWithDots = [NSMutableArray arrayWithCapacity:[
    start daysBetweenDate:end]];
    while(YES){
      if ([dates containsObject:dateIterator]) {
        [daysWithDots addObject:@YES];
      } else {
        [daysWithDots addObject:@NO];
      }
      NSDateComponents *info = [dateIterator
    dateComponentsWithTimeZone:self.monthView.timeZone];
      info.day++;
      dateIterator = [NSDate dateWithDateComponents:info];
      if([dateIterator compare:end]==NSOrderedDescending) break;
    }
  return daysWithDots;
}
```

Výpis 4: Metoda popisující algoritmus pro označení tréninkových dnů v kalendáři

5.8 Výstupy implementace

Výstupy uživatelského rozhraní pro výběr cviků a zobrazení zaznamenaných dat v kalendáři jsou znázorněny obrázkem 3. Levá část obrázku 3 reprezentuje seznam cviků s možností hledání cviků podle jeho názvu či procvičované svalové partie. Zobrazení detailních informací cviku je možné po zvolení cviku ze seznamu. Přidání vlastního cviku je možné po dotyku tlačítka plus, kdy uživateli vyjede obrazovka určující název nového cviku společně s přiřazením procvičující svalové partie. Zde je po zvolení svalové partie z další obrazovky předán objekt obrazovce předchozí. Po stisknutí tlačítka *Save* se nový cvik uloží a seznam cviků aktualizuje svůj obsah z důvodu zobrazení nového cviku. Cvik je zároveň zařazen do správné sekce pro seskupení cviků dle hlavní svalové skupiny, jenž cvik procvičuje.

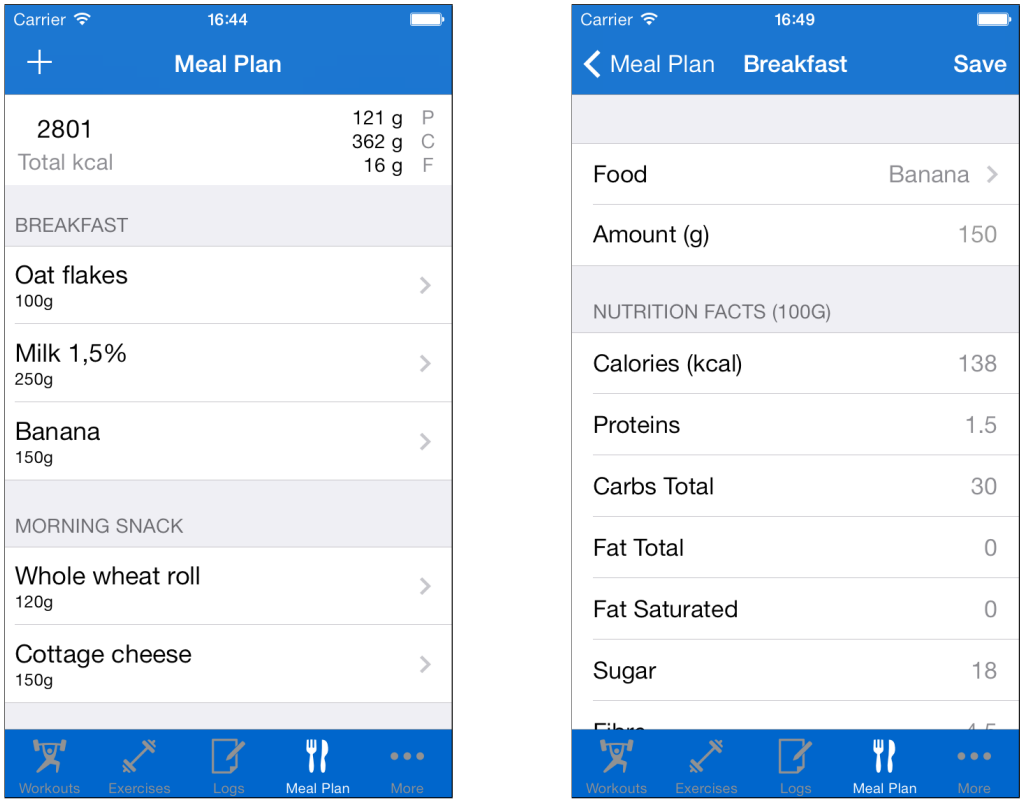
Pravá část obrázku 3 reprezentuje obrazovku, kdy je uživateli zobrazen kalendář se zaznamenanými tréninky. Pro tento účel je v aplikaci využit prvek kalendáře z *Tapku* knihovny, jejíž práci popisuje kapitola 5.7.1. Dny obsahující trénink jsou v kalendáři označeny kolečkem a jejich zobrazení určuje den, který uživatel zvolil. Po zvolení tréninku ze seznamu, nacházejícího se pod kalendářem, je uživateli zobrazen seznam odcvičených cviků. Po zvolení konkrétního cviku jsou zobrazeny informace o jednotlivých sériích, zvednuté zátěže a počtu opakování.



Obrázek 3: Výsledné uživatelské rozhraní aplikace

Jídleníček znázorňuje levá část obrázku 4. Uživateli jsou zobrazeny jídla a jejich množství seskupených dle denní doby příjmu. Je vypočítán celkový kaloriční příjem jídelníčku společně s celkovým podílem tří hlavních živin. Písmeno P značí celkový podíl bílkovin, písmeno C podíl sacharidů a písmeno F podíl tuků v jídelníčku. Tyto hodnoty jsou automaticky aktualizovány při provedených změnách v jídelníčku. Zároveň může může uživatel na této obrazovce jídla editovat přímo označením řádku. Vložení jídel je možné skrze tlačítko plus, kdy uživatel z další obrazovky zvolí denní dobu příjmu, vybere jídlo a zadá jeho množství. Toto chování popisuje pravá část obrázku 4.

Změnou množství jídla jsou ihned přepočítány jeho nutriční hodnoty, a tak jsou uživateli v reálném čase zobrazeny informace živin aniž by muselo být dané jídlo součástí jídelníčku. Tuto činnost lze provádět i při editaci jídel, které už jsou v jídelníčku obsaženy. Vytváření vlastních jídel je možné na obrazovce, jenž se zobrazí pro výběr jídla ze seznamu tlačítkem plus. Uživatel má zde možnost vložit název jídla i nutriční hodnoty nového jídla a zakomponovat ho tak do jídelníčku. Zastoupení nutričních hodnot uživatel zadává ve sto gramech potraviny.



Obrázek 4: Konečné uživatelské rozhraní aplikace pro práci s jídly

6 Závěr

Cílem bakalářské práce bylo vytvořit mobilní aplikaci pro lidi zabývající se cvičením ve fitness centrech a zdravým životním stylem a podpořit tím tak jejich snažení již při vykonávání samotného tréninku. Byl prozkoumán trh současných řešení tohoto tématu, kde vyšlo na povrch, že i když se zdravým životním stylem souvisí kromě procvičování lidského těla i zdravé stravování, tak aplikace, jenž byly součástí vyhodnocení, pokrývaly jen jednu z částí této problematiky. V souladu s uživatelskými požadavky cílové skupiny a vyhodnocením trhu bylo navrženo a implementováno řešení umožňující uživateli záznam dosažených výsledků kapesními zařízeními běžící na mobilní platformě iOS. Při implementaci byl kladen důraz na přívětivost a celkovou přehlednost uživatelského rozhraní a to tak, aby byl záznam dat při samotném tréninku efektivní a co nejpraktičtější. K uchování a synchronizaci dat byl použit framework Core Data, který v kombinaci s technologií iCloud disponuje automatickou synchronizací dat v reálném čase napříč iOS platformou, kde není potřebná žádná uživatelská režie nutná k vyvolání této akce.

Splněny jsou všechna kritéria prezentována tabulkou 1, na základě kterých byly posuzovány existující řešení. Výsledná aplikace tedy umožňuje uživateli použít aplikaci bez nutnosti registrace uživatele, vytvářet vlastní cviky i tréninky, hledat cviky na základě názvu cviku či procvičující svalové partie se zobrazením detailu cviku, záznam sérií, zobrazit dosažená data z historického hlediska, vytvořit vlastní jídla s přiřazením nutričních hodnot a následnou tvorbou jídelníčku obsahující kalkulaci celkového kalorického příjmu se zobrazením podílů tří hlavních makronutrientů. Pro zobrazení výsledků zadaných při tréninku v minulosti byla v aplikaci použita knihovna třetích stran obsahující kalendář, který není součástí vývojové sady.

Budoucí rozšíření Aplikaci plánuji rozšířit o použití systémové notifikační lišty pro informování uživatele zobrazující budoucí tréninky a grafové výstupy, které nebyly realizovány z důvodu zvýšené časové náročnosti pro studium tohoto řešení, kdy zařazení tohoto řešení není součástí vývojové sady. Bylo by tak potřeba sáhnout po další knihovně či frameworku třetích stran, kde je na každé další použití dodatečné knihovny investována značná část času celého vývoje aplikace.

Přínosy Tuto bakalářskou práci jsem si vybral z důvodu vlastního zájmu rozšíření vědomostí o okruh znalostí týkající se vývoje aplikací pro iOS platformu, který není součástí studijních plánů školy. Čtenáři této práce jsou znázorněny a předkládány principy používající se při programování pro iOS a OS X, syntaxe jazyka a celkově je tak možné nahlédnout do světa vývoje pro tyto platformy. Možná je také distribuce aplikace na App Store, tak aby z výsledků této práce mohli těžit i koncoví uživatelé se zájmem o zdravý životní styl.

7 Reference

[1] BUCK, Erik M a Donald A YACKTMAN. *Cocoa design patterns*. Upper Saddle River, NJ: Addison-Wesley, c2010, xxv, 427 s. ISBN 978-0-321-53502-3.

[2] CHUNG, Carlo. *Pro Objective-C design patterns for iOS*. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2011, xiv, 375 p. ISBN 14-302-3330-3.

[3] MARK, Dave a Jeff LAMARCHE. *iPhone SDK: průvodce vývojem aplikací pro iPhone a iPod touch*. Vyd. 1. Brno: Computer Press, 2010, 480 s. ISBN 978-80-251-2820-6

[4] KOCHAN, Stephen G. Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone. Vyd. 1. Brno: Computer Press, 2010, 550 s. ISBN 978-80-251-2654-7.

[5] Apple: Developer Portal [online]. `http://developer.apple.com/`, 2012-12 [cit. 2014-05-07].

[6] Tapku: Reference [online]. `http://devinsheaven.com/tapku/documentation/`, 2012-12 [cit. 2014-05-07].

A Aplikace

Hlavní přílohou této práce je samotná aplikace nacházející se na přiloženém kompaktním disku. Disk obsahuje archív v němž je umístěn projekt aplikace a tento text. Projekt je umístěn pod názvem `inShape.xcodeproj` a spustit jej lze až po zkompilování na počítači firmy Apple použitím IDE Xcode 5. Pro otestování iCloud synchronizace je nutné, aby byl uživatel registrovaným vývojářem.

Při návrhu aplikace byl kladen důraz na její jednoduché ovládání, a tak je uživatelská dokumentace aplikace částečně nastíněná kapitolou 5.8